Requirements Evolution

Ch. 6 Lecture Notes

IN4MTX 113

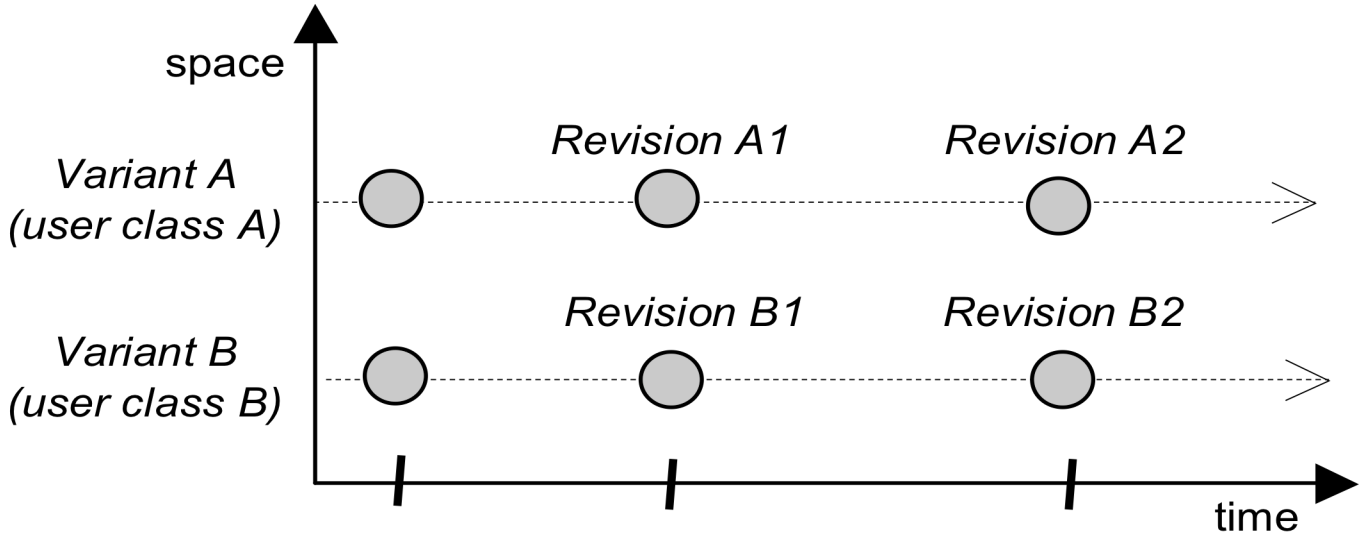
February 2010

Chapter 6 Topics

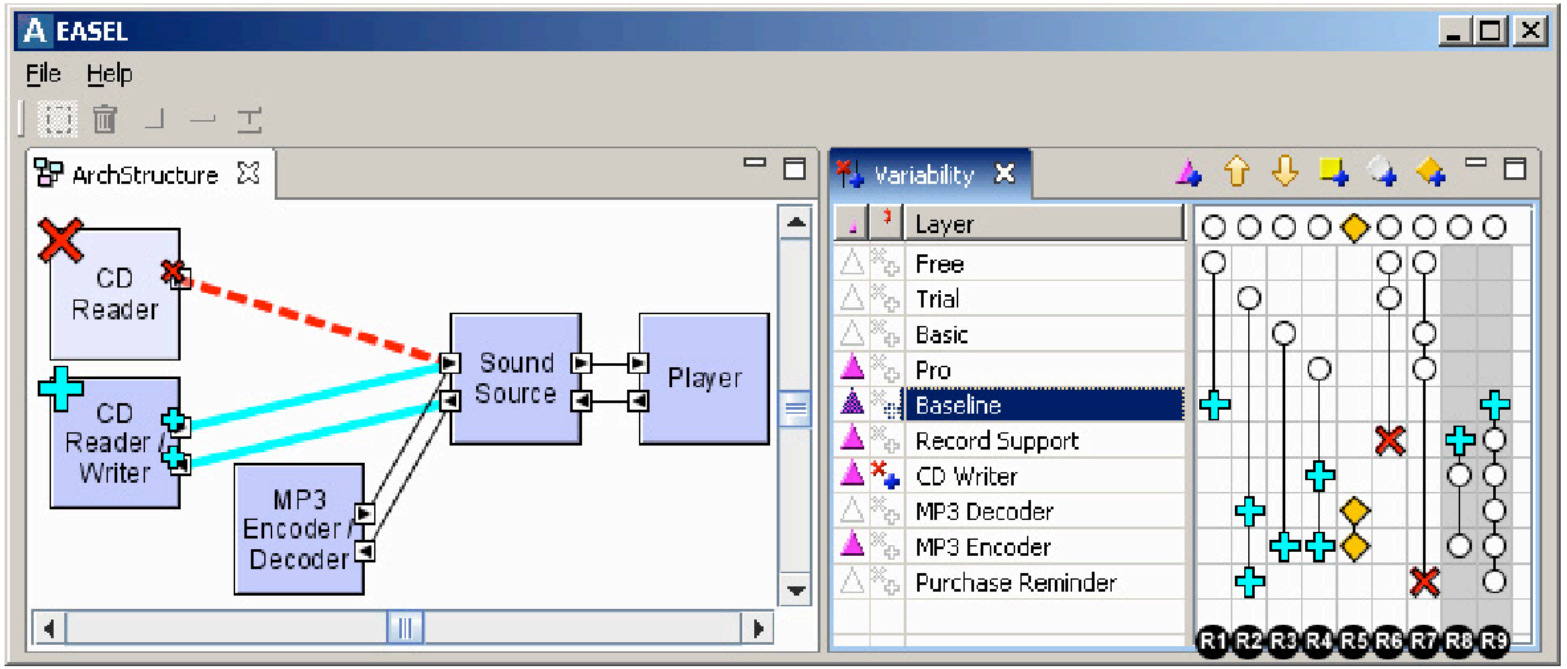
- Sources of change
- Versions (Revisions) and Variants
- Configuration Management
- Change propagation
- Trace links
- Verifiability
- Organization of a requirements document for change

Versions and Variants

- Versions: new in *time* sequence
- Variants: members of a product family having some features in common



A Simple Product Family



Configuration Management

- Version control systems (VCSs) capture “the items to be versioned, the common properties shared by all version of an item, and the deltas [and also] determines the way version sets are organized” [Clemm 2002]
  - An Extensional approaches retrieve reversions of artifacts that have been previously checked in [Conradi 1998]
  - Intensional approaches construct artifact versions based on rules describing consistent combinations [Conradi 1998]
- Example: Subversion
  - **Version Control with Subversion**

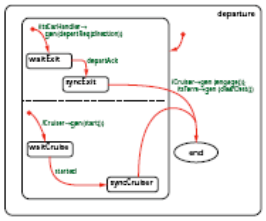
Change Propagation

- What work do you have to do if:
  - An error is found in a requirments document?
  - A requirement is found to be infeasible
    - (as a result of trying to design or program a solution)
- How does this change when managing a product family?
- Example: suppose the accelerator pedal “sticks” in a car -- what models do you recall/fix?

Trace links

Motivating Scenario

Java package X

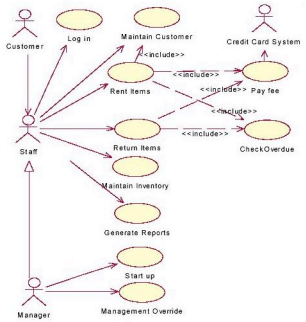


Statecharts

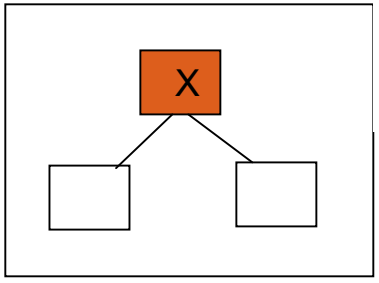
Req Spec

Func Spec

Video Rental Store Use Case Diagram

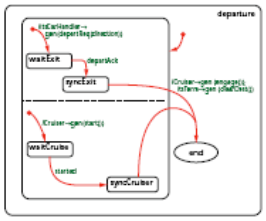


Use Cases



Structural Design

Motivating Scenario



Statecharts

Req Spec

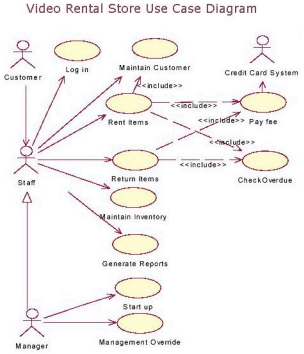
Func Spec

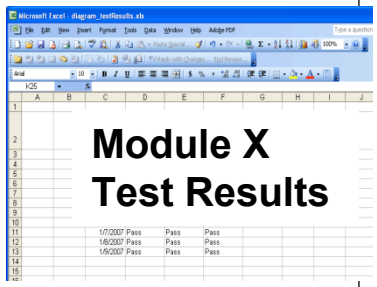
Java package X

Bug reports

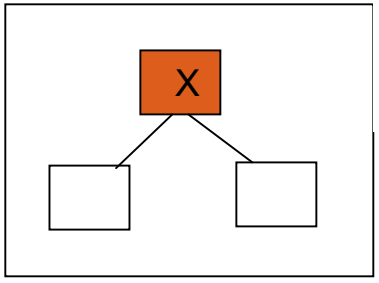
Developer Wiki

Module X



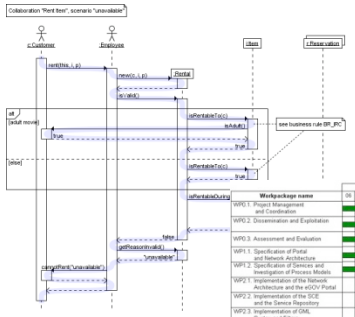


Notes

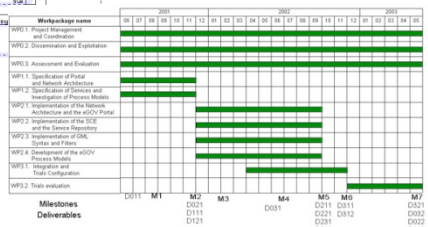


al Design

Use Cases

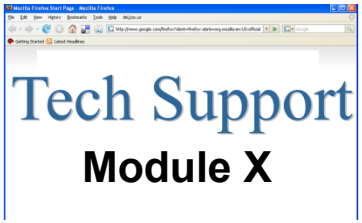


Sequence

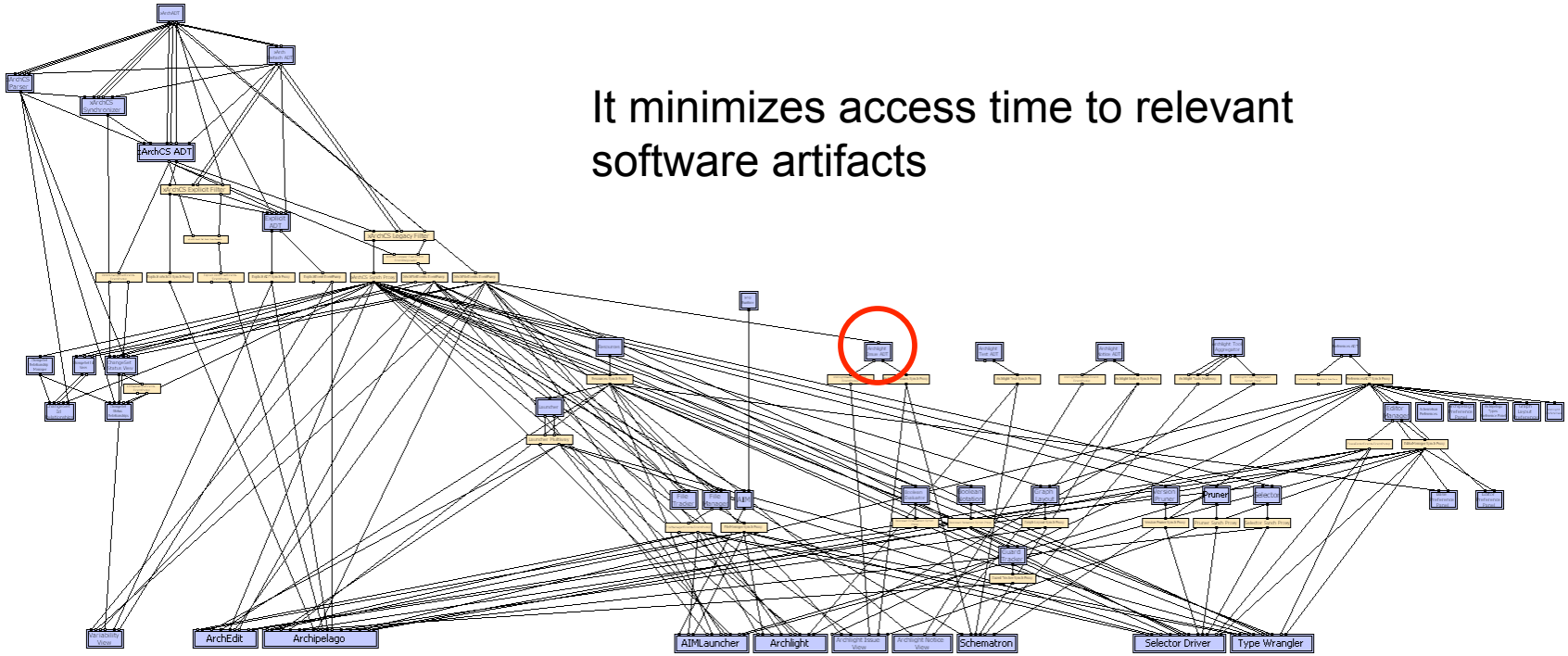


Gantt Chart

Mgmt Reports



Why Software Traceability?



It minimizes access time to relevant software artifacts

Questions

- Why is this component used? Can it be replaced?
- Who owns the related artifacts?
- Is the component tested? Does it meet requirements (if it exists)?

Traceability aids in

- System comprehension & impact analysis [Lindvall 1996, Ramesh 2001, Jarke 1998]
- Communication between stakeholders [Pohl 1994]
- System debugging [Jarke 1998, Richardson 2004]

Multi-faceted Traceability Problem

- High cost [Jarke 1998, Ramesh 2001]
- Explosion of artifact/relationship space [Domges 1998, Ramesh 2001]

Economic

- Link deterioration [Hayes 2007, Ramesh 2001]
- Heterogeneity of artifacts
  [Anderson 2002, Lindvall Practical]
- Heterogeneity of tools
  [Domges 1998, Gotel 1994, Ramesh 1995]

Technical

- Different groups
  [Gotel 1994, Ramesh 1995, Ramesh 2001]
- Different expectations
  [Gotel 1994, Ramesh 2001]
- Low motivation
  [Appleton 2005, Almeida 2006, Jarke 1998, Alexander 2002, Hayes 2005]
- Others…privacy, politics, low priority, lack of time…
  [Domges 1998, Gotel 1994, Jarke 1998, Ramesh 1995, Lindval 1996]

Social

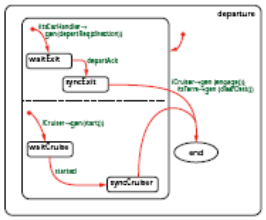
Current approaches

- Information Retrieval Techniques
  - Automatically generated links have limited link semantics [Spanoudakis 2005]
  - Usually require pre- & post-processing [Cleland-Huang 2007]
- Software Repository Mining
  - Specialized searches & retrospective capture of links [Zimmerman 2004, Cubranic 2005]
- Design Rationale
  - Difficult to retrieve & use [Horner 2005]
    - Compendium [Shum 2006] – capturing rationale requires significant effort
  - ARM [Tang 2005] caters mainly to architects and assumes the existence of requirements prior to design
- Current Link Technology
  - XML Topic Maps enable manual capture of links

Insights

- Experience in building an industrial software traceability tool
- Software Architecture
- Open Hypermedia
- e-Science

ACTS



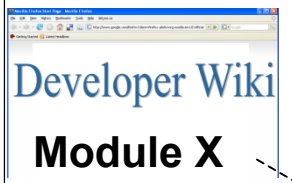
Statecharts

Local net

Req Spec

Architect perspective

Test Suites

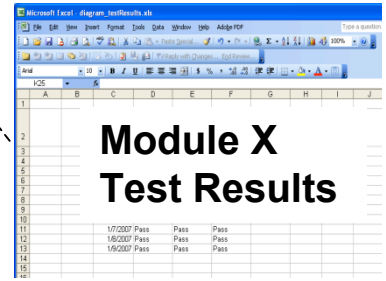


Module X

Developer perspective

QA perspective

Field engineer perspective



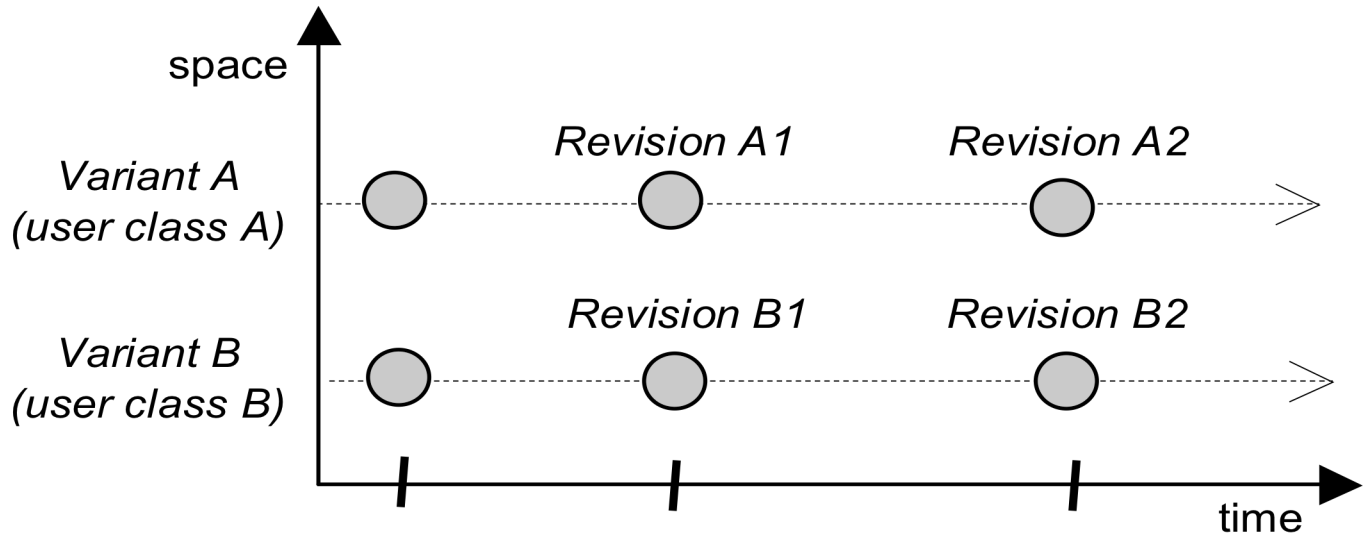
Java package X

Bug reports

Internet

Internet

Chapter 6


space
Variant A
(user class A)
Revision A1
Revision A2
Variant B
(user class B)
Revision B1
Revision B2
time

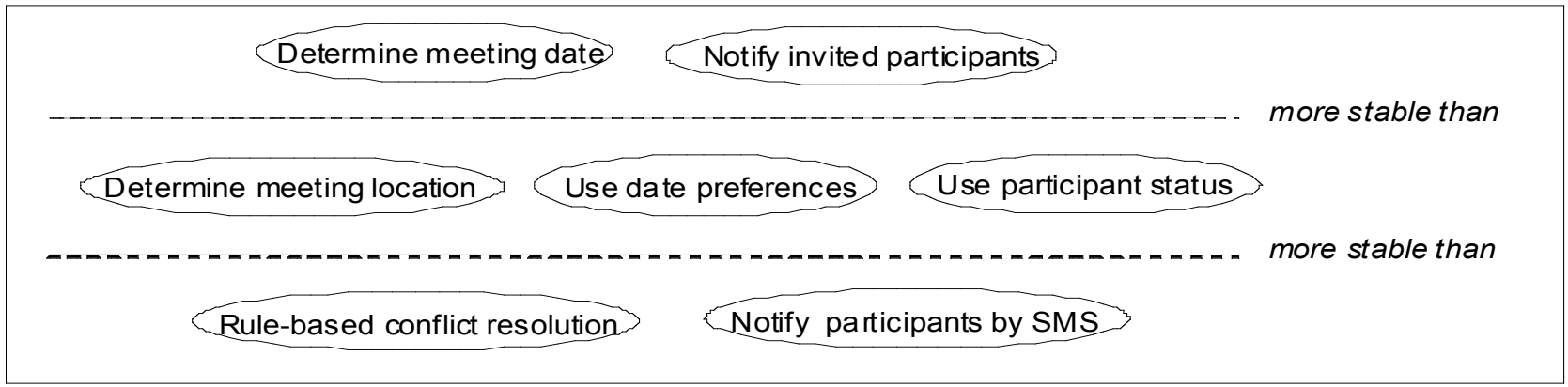


Figure 6.2 – Ordering features by levels of stability or commonality

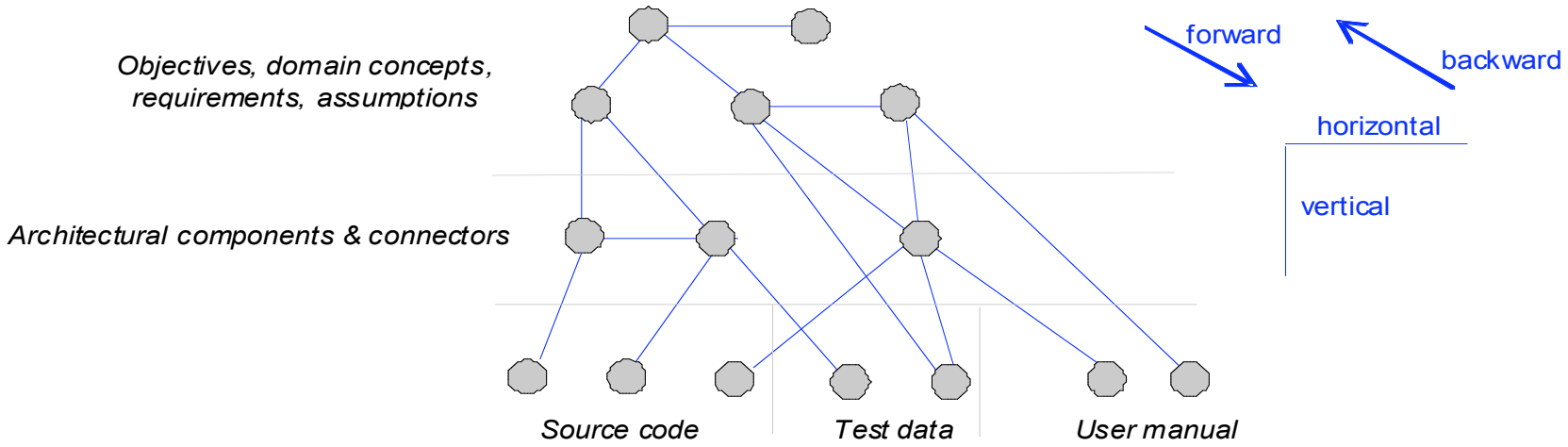


Figure 6.3 – Traceability links: forward, backward, horizontal, and vertical traceability

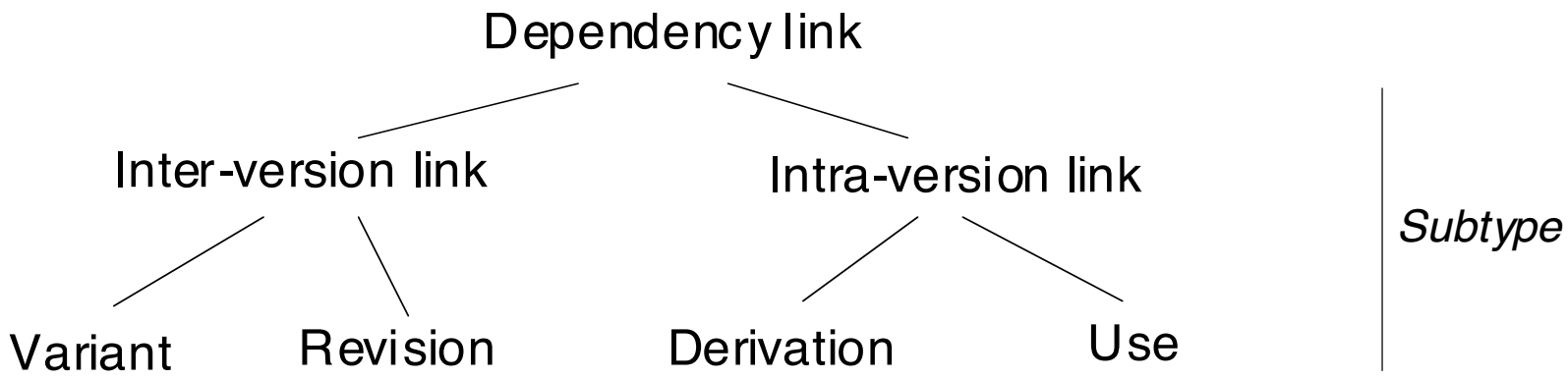


Figure 6.4 – A taxonomy of traceability link types

Figure 6.5 – *Dependency* link type

Figure 6.6 – *Variant* link type

Figure 6.7 – *Revision* link type

Figure 6.8 – *Use* link type

Figure 6.9 – *Derivation* link type

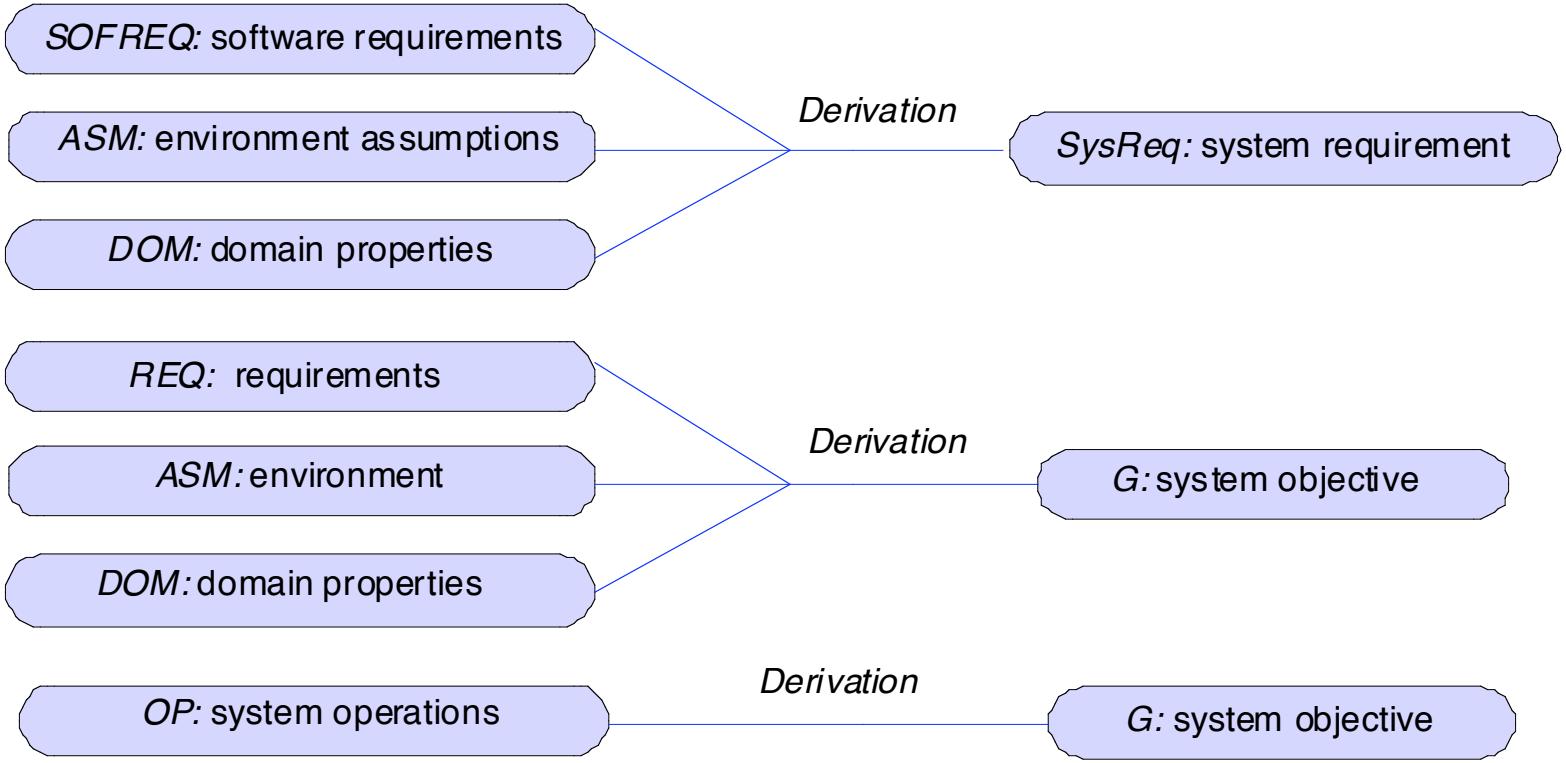


Figure 6.10 – Derivational traceability links implied by satisfaction arguments

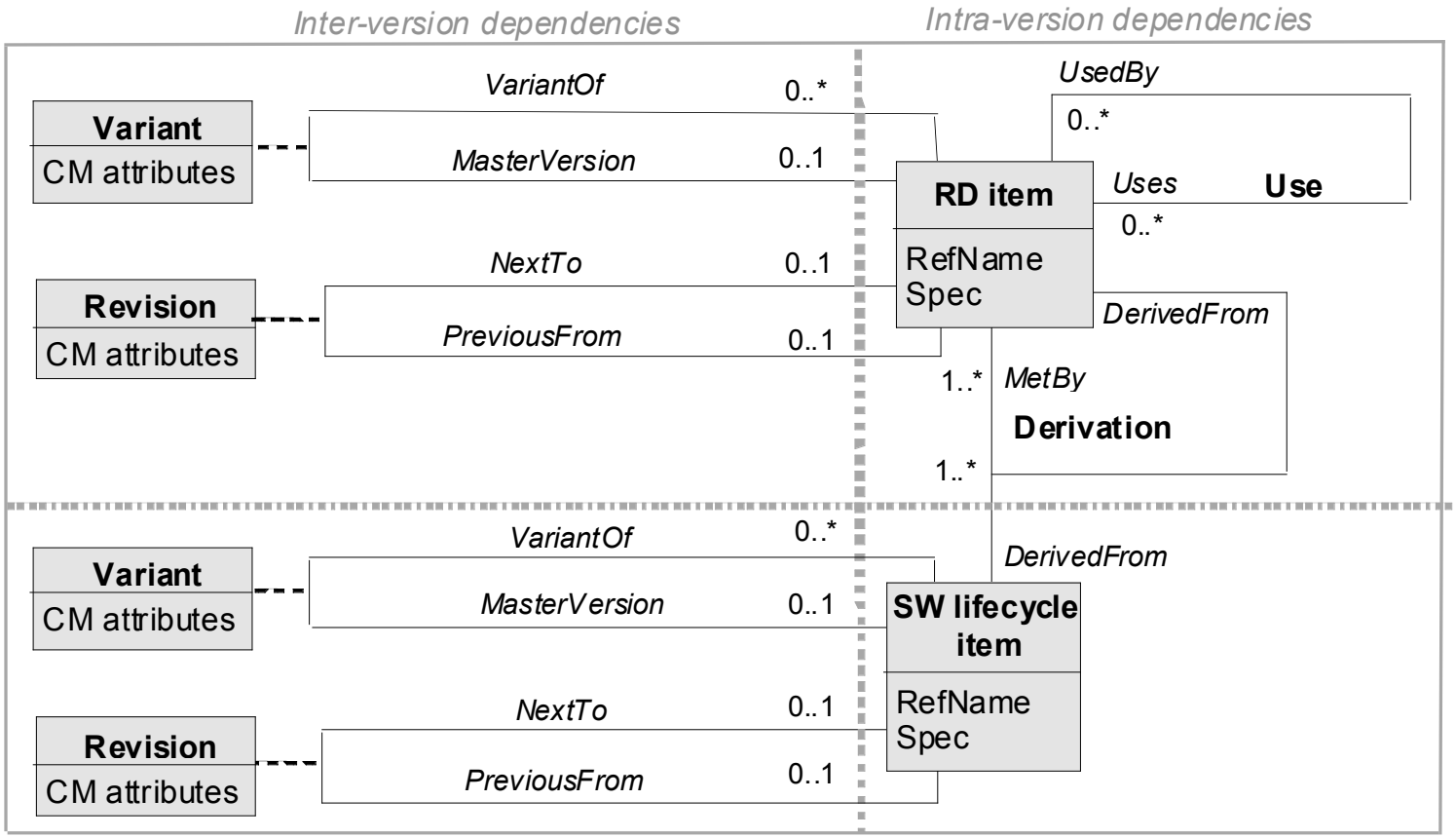


Figure 6.11 – Item traceability: an entity-relationship model

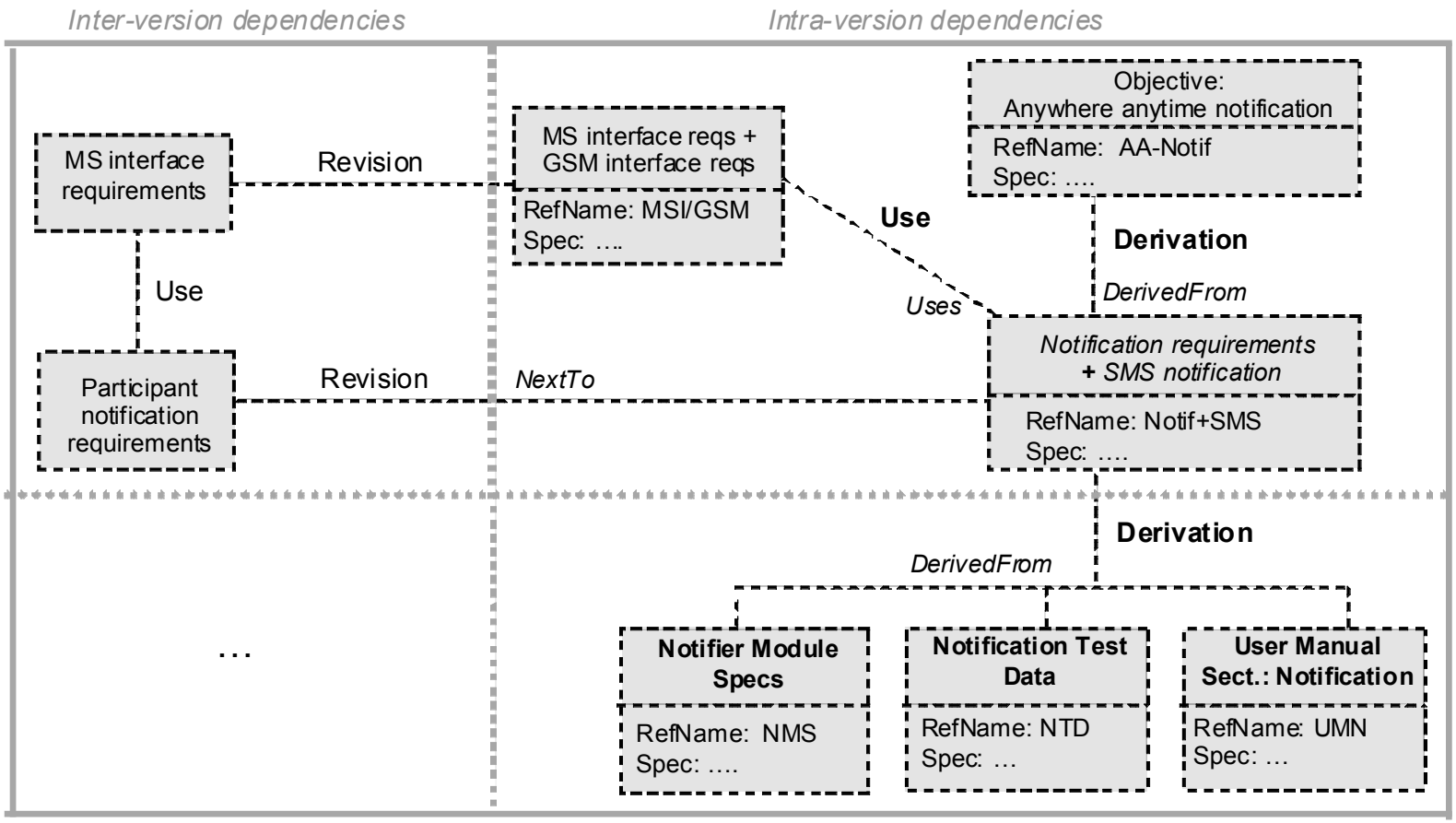


Figure 6.12 – Item traceability: model instantiation to meeting scheduling

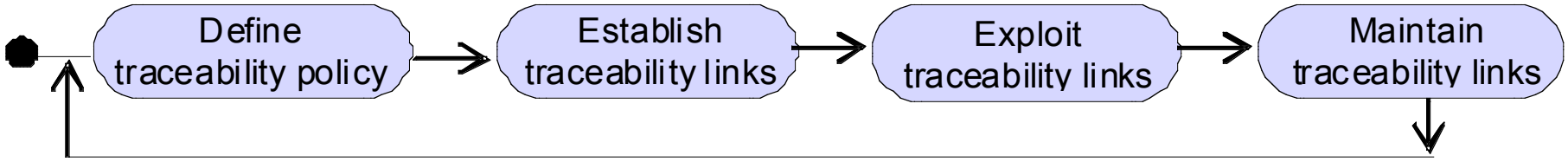


Figure 6.13 – Traceability management

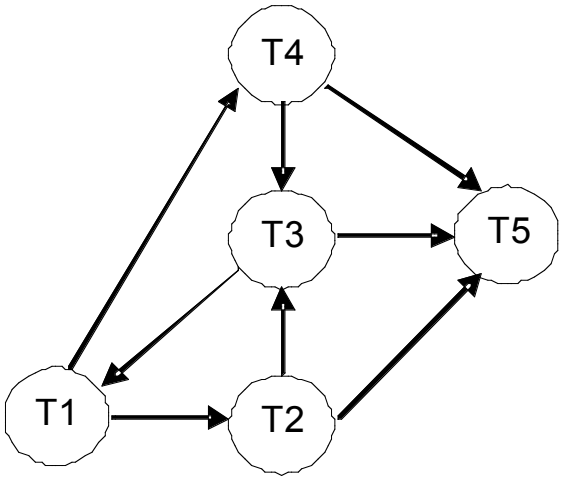


Figure 6.14 – Single-relation traceability graph represented by the matrix in Table 6.2

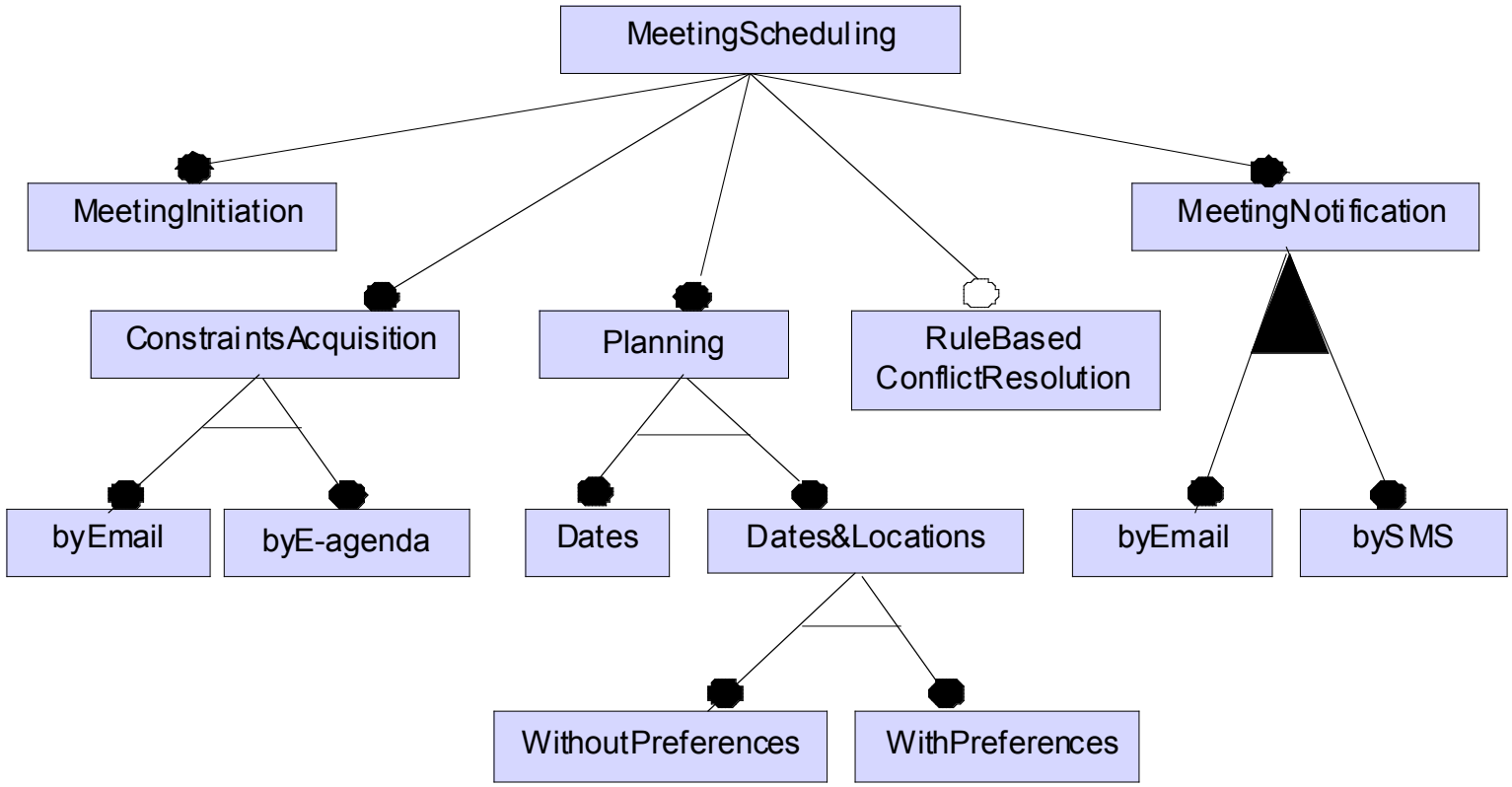


Figure 6.15 – Feature diagram for variants of the meeting scheduling system

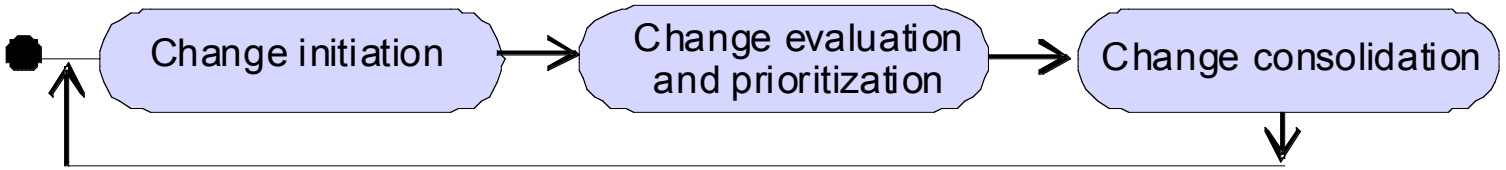


Figure 6.16 – Change control